CLAIMS

What is claimed is:

1. A method for detecting memory leaks in a software program, said method comprising the steps of:

monitoring a specified one or more analysis properties of software objects executing in the software program, wherein the one or more specified analysis properties consists of at least one of an object's age and an object's instance count;

determining if any analysis property of software objects being referenced following a garbage collection process exceeds a respective predetermined limit for such analysis property, wherein a predetermined limit for an object's age is an object age limit and a predetermined limit for an object's instance count is an object instance count growth value; and

identifying any software objects determined to have one or more analysis properties that exceeds that property's predetermined limit.

2. The method according to claim 1, further comprising the step of calculating an object's age by timing a current period starting when the respective object was instantiated.

3. The method according to claim 1, further comprising the step of calculating object instance count growth as the magnitude of growth of an object's instance count over a given time period.

4. The method according to claim 1, wherein the step of monitoring comprises monitoring objects within a class designated for monitoring.

5. The method according to claim 1, further comprising the step of performing a stack walkback for the identified software objects.

6. The method according to claim 1, further comprising the step of generating a statistics report comprising the identified software objects.

7. The method according to claim 6, further comprising the step of generating a statistics report including stack walkbacks for the identified software objects.

8. The method according to claim 6, further comprising the step of generating a web interface for user viewing of the statistics report at a computer display.

9. The method according to claim 1, wherein the software objects are Java objects.

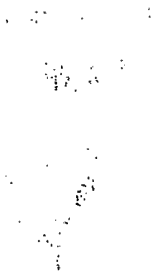
10. The method according to claim 1, further comprising the steps of:

monitoring an amount of available memory for a software program referencing software objects;

determining when the amount of available memory for the software program referencing software objects is within a predetermined threshold amount of memory within zero memory available for the software program utilizing software objects; and

upon such determination, storing a current stack walkback of currently referenced software objects prior to the amount of available memory for a software

program referencing software objects dropping below an amount of available memory necessary to store a current stack walkback.



11. A system for detecting memory leaks in a software program comprising:

means for monitoring a specified one or more analysis properties of software objects executing in the software program, wherein the one or more specified analysis properties consists of at least one of an object's age and an object's instance count;

means for determining if any analysis property of software objects being referenced following a garbage collection process exceeds a respective predetermined limit for such analysis property, wherein a predetermined limit for an object's age is an object age limit and a predetermined limit for an object's instance count is an object instance count growth value; and

means for identifying any software objects determined to have one or more analysis properties that exceed that property's predetermined limit.

12. The system according to claim 11, further comprising means for calculating an object's age by timing a current period starting when the respective object was instantiated.

13. The system according to claim 11, further comprising means for calculating object instance count growth as the magnitude of growth of an object's instance count over a given time period.

14. The system according to claim 11, wherein the means for monitoring comprises monitoring objects within a class designated for monitoring.

15. The system according to claim 11, further comprising means for performing a stack walkback for the identified software objects.

16. The system according to claim 11, further comprising means for generating a statistics report comprising the identified software objects.

17. The system according to claim 16, further comprising means for generating a statistics report including stack walkbacks for the identified software objects.

18. The system according to claim 16, further comprising means for generating a web interface for user viewing of the statistics report at a computer display.

19. The system according to claim 11, wherein the software objects are Java objects.

20. The system according to claim 11, further comprising:

means for monitoring an amount of available memory for a software program referencing software objects;

means for determining when the amount of available memory for the software program referencing software objects is within a predetermined threshold amount of memory within zero memory available for the software program utilizing software objects; and

means for, upon such determination, storing a current stack walkback of currently referenced software objects prior to the amount of available memory for a software program referencing software objects dropping below an amount of available memory necessary to store a current stack walkback.

21. An article of manufacture comprising machine-readable medium including program logic embedded therein for detecting memory leaks in a software program that causes control circuitry in a data processing system to perform the steps of:

monitoring a specified one or more analysis properties of software objects executing in the software program, wherein the one or more specified analysis properties consists of at least one of an object's age and an object's instance count;

determining if any analysis property of software objects being referenced following a garbage collection process exceeds a respective predetermined limit for such analysis property, wherein a predetermined limit for an object's age is an object age limit and a predetermined limit for an object's instance count is an object instance count growth value; and

identifying any software objects determined to have one or more analysis properties that exceeds that property's predetermined limit.

22. The article of manufacture of Claim 21, further comprising the step of calculating an object's age by timing a current period starting when the respective object was instantiated.

23. The article of manufacture of Claim 21, further comprising the step of calculating object instance count growth as the magnitude of growth of an object's instance count over a given time period.

24. The article of manufacture of Claim 21, wherein the step of monitoring comprises monitoring objects within a class designated for monitoring.

25. The article of manufacture of Claim 21, further comprising the step of performing a stack walkback for the identified software objects.

26. The article of manufacture of Claim 21, further comprising the step of generating a statistics report comprising the identified software objects.

27. The article of manufacture of Claim 26, further comprising the step of generating a statistics report including stack walkbacks for the identified software objects.

28. The article of manufacture of Claim 26, further comprising the step of generating a web interface for user viewing of the statistics report at a computer display.

29. The article of manufacture of Claim 21, wherein the software objects are Java objects.

30. The article of manufacture of Claim 21, further comprising the steps of:

monitoring an amount of available memory for a software program referencing software objects;

determining when the amount of available memory for the software program referencing software objects is within a predetermined threshold amount of memory within zero memory available for the software program utilizing software objects; and

upon such determination, storing a current stack walkback of currently referenced software objects prior to the amount of available memory for a software program referencing software objects dropping below an amount of available memory necessary to store a current stack walkback.